1Q-05

ダブル配列法の記憶領域圧縮手法

泓田 正雄　森田 和宏　住友 徹　青江 順一
徳島大学工学部知能情報工学科

1 はじめに

情報検索技術では，検索の高速化や辞書の省メモリ化という必須の条件が要求されている．自然言語処理における辞書の検索手法としてトライ法が広く使われており [1]，トライ法を表現するデータ構造としては，ダブル配列法が検索の高速性とコンパクト性において非常に優れている [2][3]．トライ法に関してはトライを構成する状態数を削減することにより圧縮を試みる手法が多く研究されているが，実装法により記憶量を圧縮する研究はあまり行なわれていない．

そこで，本論文では「検索の高速性」「コンパクト性」という2つの特徴を併せ持つダブル配列を対象に，プログラムでの実装を考慮して記憶量の圧縮を実現する分割ダブル配列法を提案する．本手法では，一つの大きなトライを複数の小さいトライに分割することにより，各トライの最大状態番号を小さくする．最大状態番号が小さくなれば，状態番号を表すために必要なビット数が少なくなるので，記憶量の圧縮を行うことができる．

2 ダブル配列

トライを表現するデータ構造としては，配列構造やリスト構造が広く知られている．青江の提案したダブル配列法は，トライを二つの配列構造で表現し，配列構造の高速性とリスト構造のコンパクト性の両方の特徴を持つ優れたデータ構造である．

ダブル配列法では，まず，キー中の各文字に内部表現値と呼ばれる値を設定する．そして，文字 c に対する内部表現値を CODE(c) としたとき，状態 s から状態 t へ文字 c によって遷移が可能であれば，

t =BASE[s]+CODE(c)

CHECK[t]= s

の二式を満足する．1式目により，状態 s から状態 t への遷移が決定し，2式目により，状態 t が状態 s からの遷移であることを確認している．ダブル配列の状態番号は BASE，CHECK の要素番号に対応しており，この状態番号はあらかじめ決定しておくのではなく，キーの登録時に動的に決定される．

A Compression Method of Double Array Structures
Masao Fuketa, Kazuhiro Marita,
Toru Sumitomo and Jun-ichi Aoe
Department of Information Science and Intelligent Systems,
University of Tokushima

しかし，トライでは，登録するキーが多くなればなるほど，状態数が増えるので，状態番号を表すために必要なビット数も増えてしまう．状態番号を4バイトで表せば，約20億までの値を用いることができるので，大規模なキー集合を表すことができるが，ダブル配列法では一つの状態を表現するのに BASE に4バイト，CHECK に4バイトの合計8バイトが必要となるので，消費記憶領域が大きくなってしまう．

3 分割ダブル配列

状態番号を表すために必要なビット数を少なくするために，一つの大きなトライを複数の小さなトライ（ブロック）に分割する手法を提案する．トライの分割により，一つ一つのトライの最大状態番号は小さくなるので，ビット数を少なくできるが，分割したトライを遷移する必要が出てくる．この分割された状態を繋ぐために TABLE を用意する．この TABLE は遷移先ブロックを示す *block* と遷移先ブロックでの状態番号を示す *state* を持ち，どの状態に遷移しているかを一意に決定できる．例えば，図1では，ブロック1の状態7からブロック2の状態1に遷移することを表している．

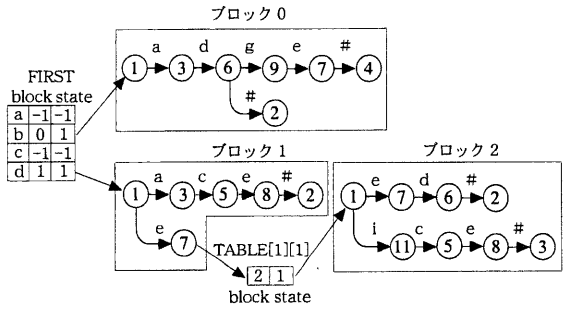

図 1: 分割トライ

分割の操作を繰り返すことにより，各トライの最大状態番号は小さくなる．しかし，複数のブロックに跨っているキーを検索する場合，TABLE を参照しなければならないので，検索速度が低下してしまう．この問題を最小限にする為には，なるべく多くのキーを一つのブロックに格納する．つまり，TABLE を参照せずに検索可能なキーを多くすればよい．通常トライは状態番号1から検索されるので，分割す

るだけでは，最初のブロックを必ず遷移する必要があるので，TABLEを参照せずに検索できるキーはどうしても少なくなってしまう．そこで，検索キーの第一文字cによって検索開始ブロックと状態番号を，FIRST[c].*block* と FIRST[c].*state* を用いて決定する構造体FIRSTを導入する．図1で，文字'd' から始まるキーはブロック1の状態番号1から格納されているので，FIRST['d'].*block* = 1， FIRST['d'].*state* = 1となる．図2に図1のダブル配列表現を示す．この分割ダブル配列上で，他のブロックへ遷移しているとき，

S_CHECK[*block*][*state*]< 0

とし，S_BASE[*block*][*state*] の値はTABLEのインデックスを格納する．

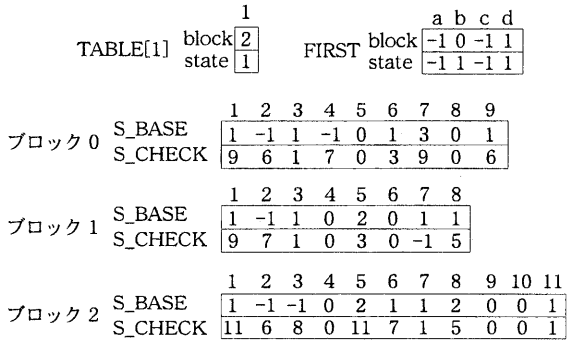

図 2: 分割ダブル配列

4 実験による評価

実験プログラムは，約1,500行のC言語で記述されており使用マシン (celeron 500MHz)，OS(Micro soft Windows NT version4.0) 上で稼動している．4種類の大規模なキー集合に対して，従来のダブル配列と分割ダブル配列でそれぞれ構成したトライを対象に，記憶領域，検索速度の比較を行う．状態番号を半端なビット数で表すと，遷移先の状態番号を求めるときに，先ずビット演算を行う必要があるので検索速度が低下してしまう．そこで，今回の実験は状態番号を，従来のダブル配列では4バイト，分割ダブル配列では2バイトとして行った．

実験には，以下のキー集合を用いた．

KEY1：日本の県市町村字名
KEY2：分類語彙表 [4]，見出し語
KEY3：EDR 英語単語辞書 [5]，見出し語
KEY4：EDR 日本語単語辞書 [5]，見出し語

キー集合の情報，分割ダブル配列の情報，総状態数と記憶領域と検索速度の比較の結果を表1に示す．総状態数はを比較すると，状態数の増加する割合は，0.005%～0.1%程度であり，増分は少ないと言える．記憶量は，分割ダブル配列は従来のダブル配列と比較して，キー集合によらず約50%になっている．

表 1: 実験結果

	KEY1	KEY2	KEY3	KEY4
キー総数	71,316	76,447	134,054	235,810
分割ダブル配列情報				
TABLE数	35	44	32	1,312
最大TABLE使用回数	1	1	1	1
総状態数				
従来法	273,984	275,891	564,313	905,855
分割ダブル配列	274,018	275,934	564,344	907,166
記憶領域				
従来法 (MByte)	2.19	2.21	4.51	7.25
分割ダブル配列 (MByte)	1.09	1.10	2.26	3.63
比率 (%)	49.8	49.8	50.1	50.1
検索速度 (ms/1000キー)				
従来法	24.98	22.94	31.04	26.41
分割ダブル配列	24.64	22.57	30.73	26.50

検索速度は，登録した全てのキーを検索した時間を計測し，1000キー辺りの平均検索時間を求めた．両手法の検索速度は，ほぼ同等の結果となった．TABLE数は，登録するキーの個数が多くなると増加してしまうが，一つのキーに対するTABLEの参照回数が最大でも1回であるので，従来法とほぼ同じ計算速度となっている．

5 結論

本発表では，キー検索手法の一手法であるダブル配列法について，一つの大きなトライを複数の小さいトライに分割することにより，記憶量の圧縮を実現する分割ダブル配列法を提案した．そして，実験により本手法の有効性を示した．今後はさらに大規模なキー集合に対して本手法の有効性を確認する．

参考文献

[1] 青江順一: キー検索技法－トライ法とその応用，情報処理, Vol. 34, pp. 1244–251 (1993).

[2] Aoe, J.: An Efficient Digital Search Algorithm by using a Double-Array Structure, *IEEE Transactions on Software Engineering*, Vol. SE-15, No. 9, pp. 1066–1077 (1989).

[3] 永田昌明: 岩波講座言語の科学〈3〉単語と辞書，岩波書店, chapter 2 (1997).

[4] 国立国語研究所: 分類語彙表（増補版） (1997).

[5] (株) 日本電子化辞書研究所: ＥＤＲ電子化辞書 (1996).